Pegasus 5: An Automated Pre-Processor for Overset-Grid CFD

Stuart Rogers
NASA Advanced Supercomputer Division
NASA Ames Research Center
http://people.nas.nasa.gov/~rogers/home.html
stuart.e.rogers@nasa.gov

Overset Short Course
September 20, 2010

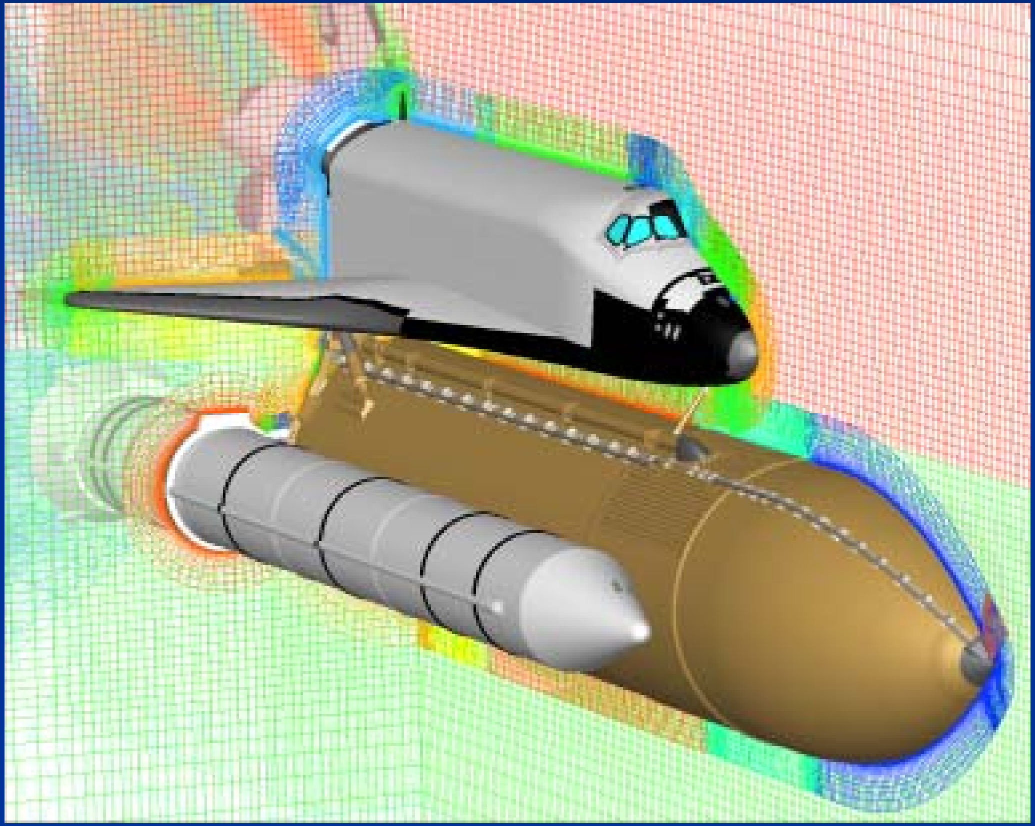
Acknowledgements

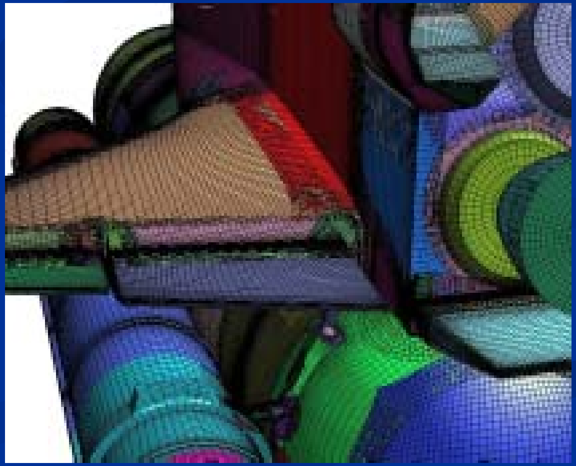
- **Pegasus 5 Primary Authors:**
  Norman Suhs, William Dietz, Stuart Rogers
- **Contributing Authors:**
  Steve Nash, William Chan, Robert Tramel, Jeff Onufer
- Developed with funding from:
  - NASA/Boeing/McDonnell-Douglas Advanced Subsonic Transport Program
  - NASA Information Power-Grid Program
  - NASA Space Shuttle Program

Outline

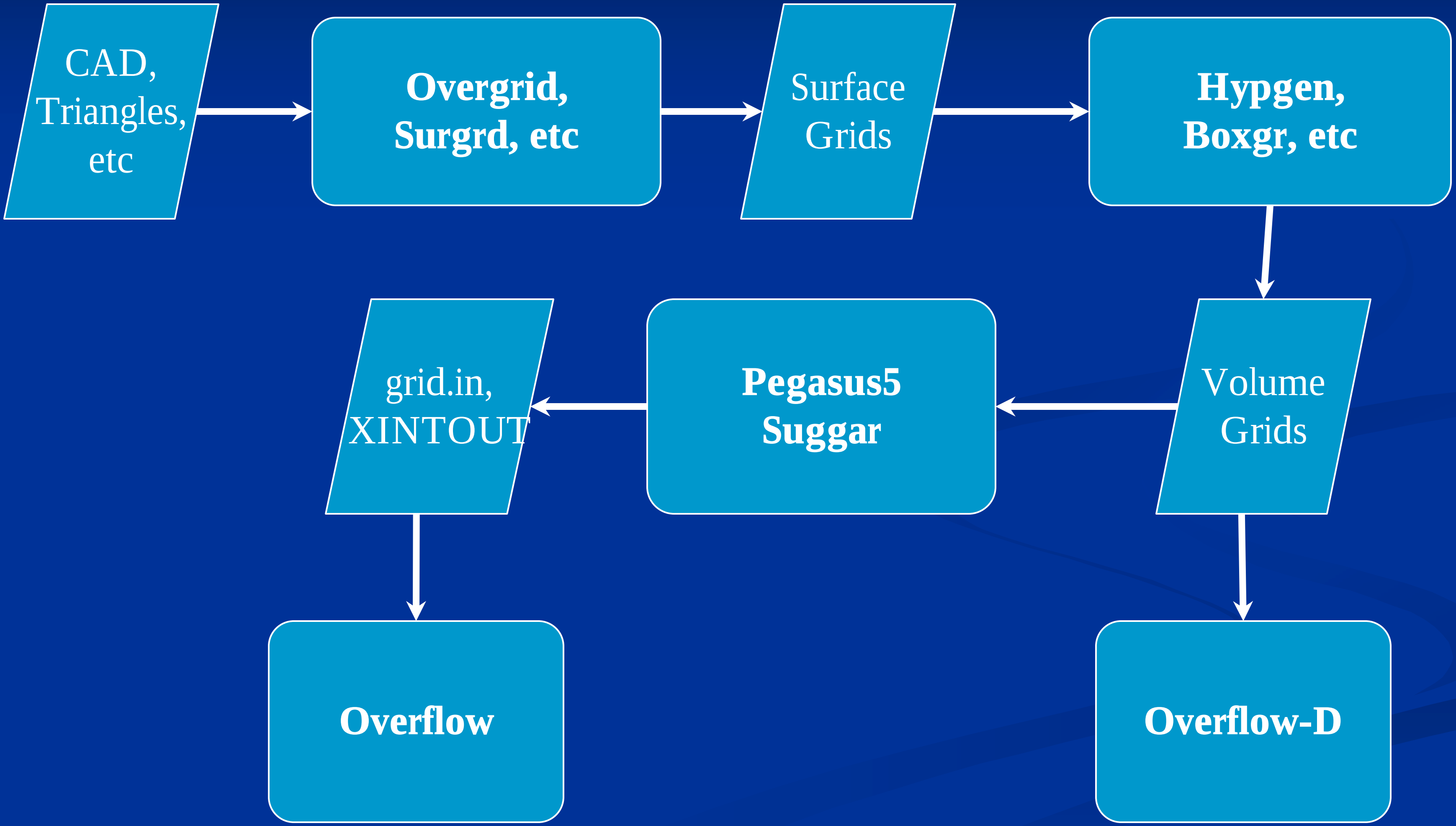
- Understanding Overset-grid work flow
- Pegasus5 features and automation
  - Auto hole cutting
  - Interpolation and overlap optimization
  - Projection
  - Restarting
  - Parallelization
- Overview of Usage
  - Required input
  - Basic Usage
  - Understanding the output
  - Overcoming problems

The Oversetting Challenge





Overset-CFD Workflow



Pegasus5 Goals and Features

- Fifth-generation overset software
  - Written in 1998-2000 as a replacement for *PEGSUS4*
- Primary goal: complete automation of overset process
  - Complexity of CFD problems continues to grow
  - Hundreds of overset zones, tens of millions of grid points
  - Manual control of process became intractable
- Required all-new approach to:
  - Hole-cutting
  - Overset optimization
- Required significant improvements in ease of use
  - Parallelization
  - Automatic restarts
  - Projection
- Maintained many *PEGSUS4* features, allowing manual control where needed
- Pegasus5 is mostly automated compared to previous generation, but still requires user knowledge and expertise

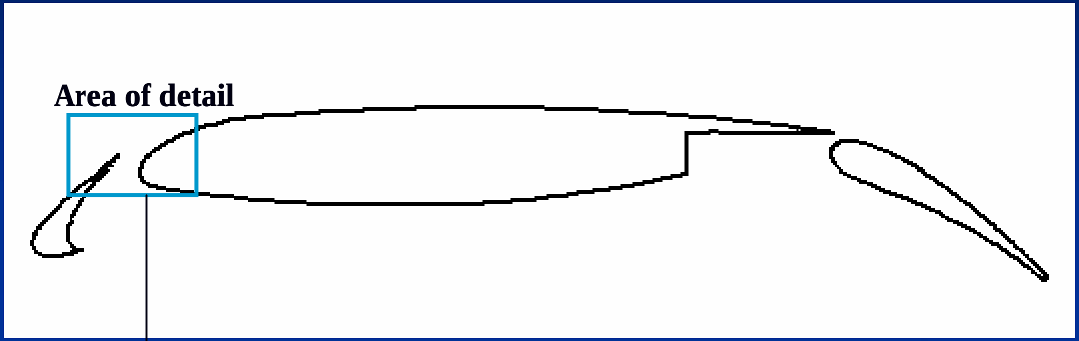
Pegasus5 Approach

- Use an Overflow-like namelist input file
  - Parsing the flow-solver boundary conditions provides most of required inputs about geometry and topology
- Use Fortran90
  - Extensive use of defined-type data and modules
  - Extensive use of process templates and data templates
- Oversetting task broken into discrete tasks
  - Input to each task saved as one or more files
  - Result of each task saved as one or more files
  - Facilitates parallelization
  - Enables restarts from partial or aborted run
  - Enables rapid restarts after change to inputs
- Use lots of computer time and lots of disk I/O
  - Intelligent use of brute-force can solve lots of issues

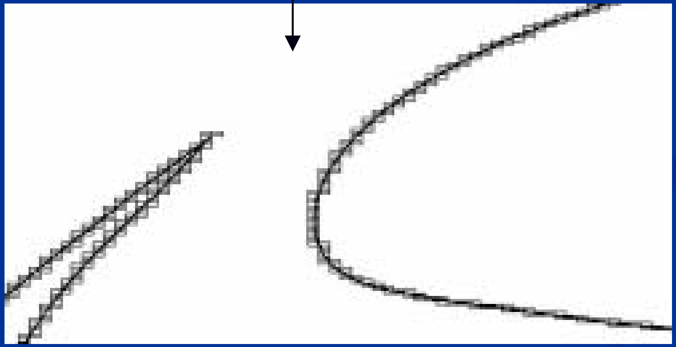
Auto-Hole Cutting Uses a Cartesian Map

Example: Multi-element Airfoil

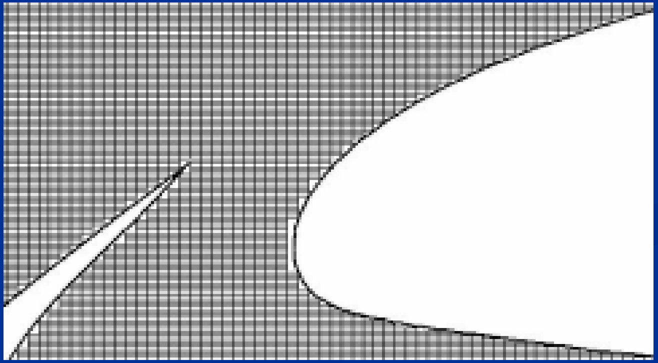
1. Identify the airtight solid-wall surfaces and overlay with a Cartesian map



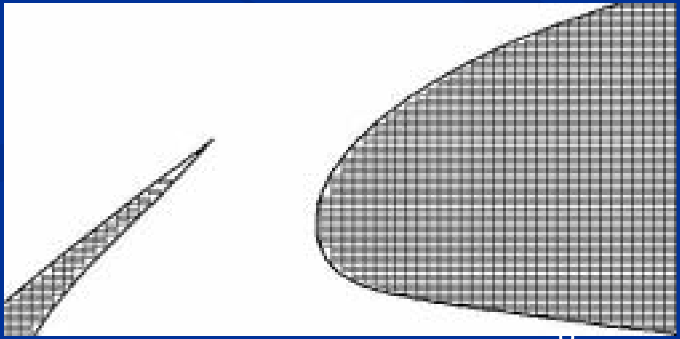
2. Find **Fringe elements**: any Cartesian element which intersects the solid wall



3. Use painting algorithm to identify **Outside elements**



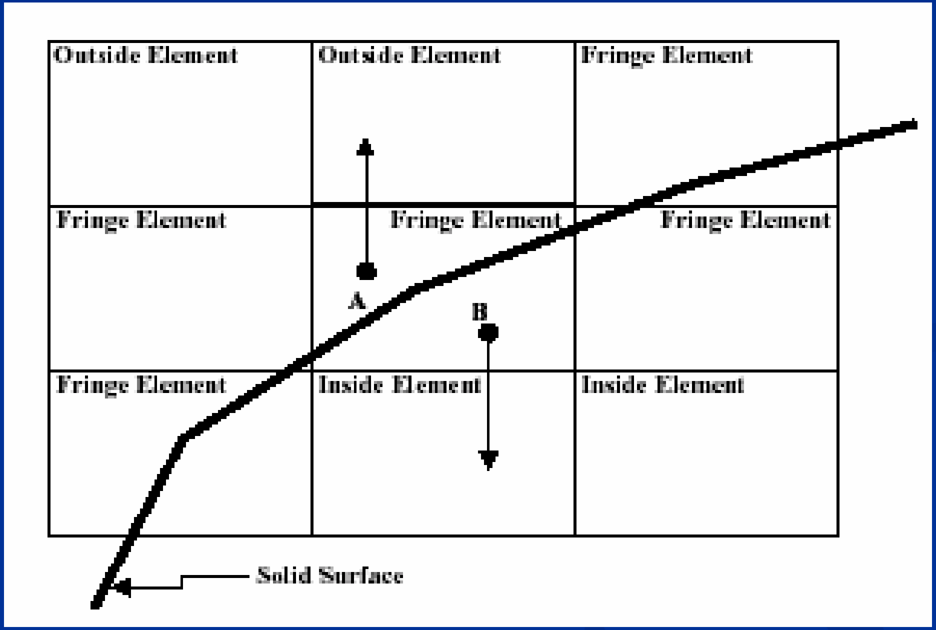
4. All others are **Inside Elements**



Auto Hole Cutting:
Cutting of Candidate Points

All volume grid points are tested as potential hole points:

- Points outside the Cartesian map are not hole points
- Points contained in an **Outside Element** are not hole points
- Points contained in an **Inside Element** are hole points
- Points contained in a **Fringe Element** undergo line-of-sight test
  - Point A can "see" an Outside Element without crossing the solid surface: it is not a hole point
  - Point B can "see" an Inside Element without crossing the solid surface: it is a hole point



Interpolation Stencil Search

- Pegasus5 searches for all possible interpolation stencil donors from all zones **for every single grid point**
  - Alternating Digital Tree (ADT) is created for all zones
  - For a given grid point and a given donor zone, an ADT lookup provides a near-by cell in donor zone, then a stencil-jumping approach finds the exact donor cell and interpolation stencil
  - All possible donor cells for every single grid point are stored

Fringe Point Identification

- A fringe point is one which requires updating in the flow-solver via interpolation from a neighboring zone
- Outer-boundary fringe points
  - All points on the boundary of a zone that do not receive a flow-solver boundary condition is identified as an outer-boundary fringe point
  - Single or double outer-boundary fringes can be requested
- Hole-boundary fringe points
  - Points adjacent to a hole point are identified as hole-boundary fringe points
  - Single or double hole-boundary fringes can be requested

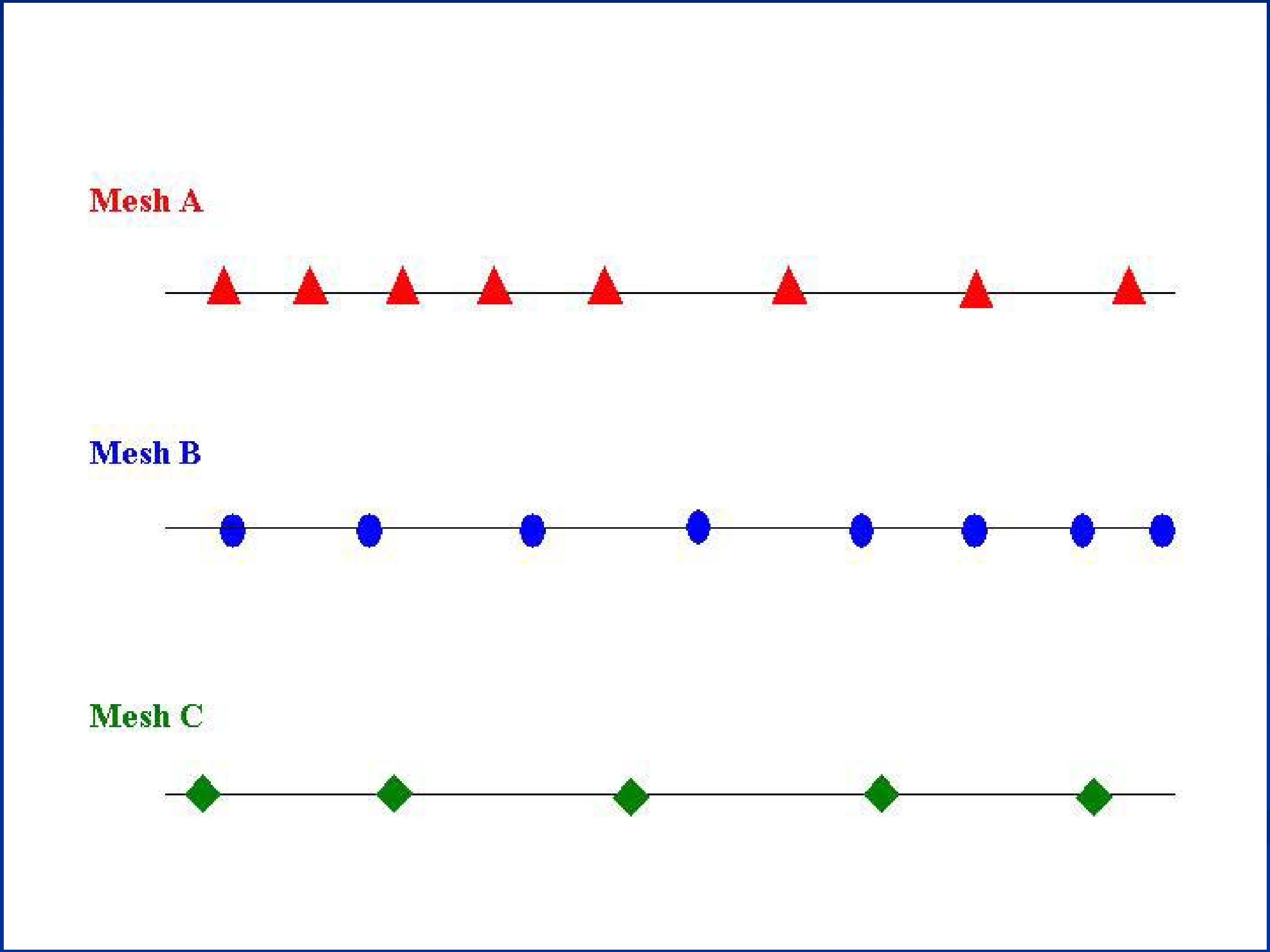
Level-1 Interpolation

- For each fringe point, the best possible interpolation stencil is chosen amongst all valid donor cells
  - When multiple donors are available, selection is based on measure of interpolation quality and relative cell size
- Any fringe point which does not have a valid donor is denoted as an orphan point

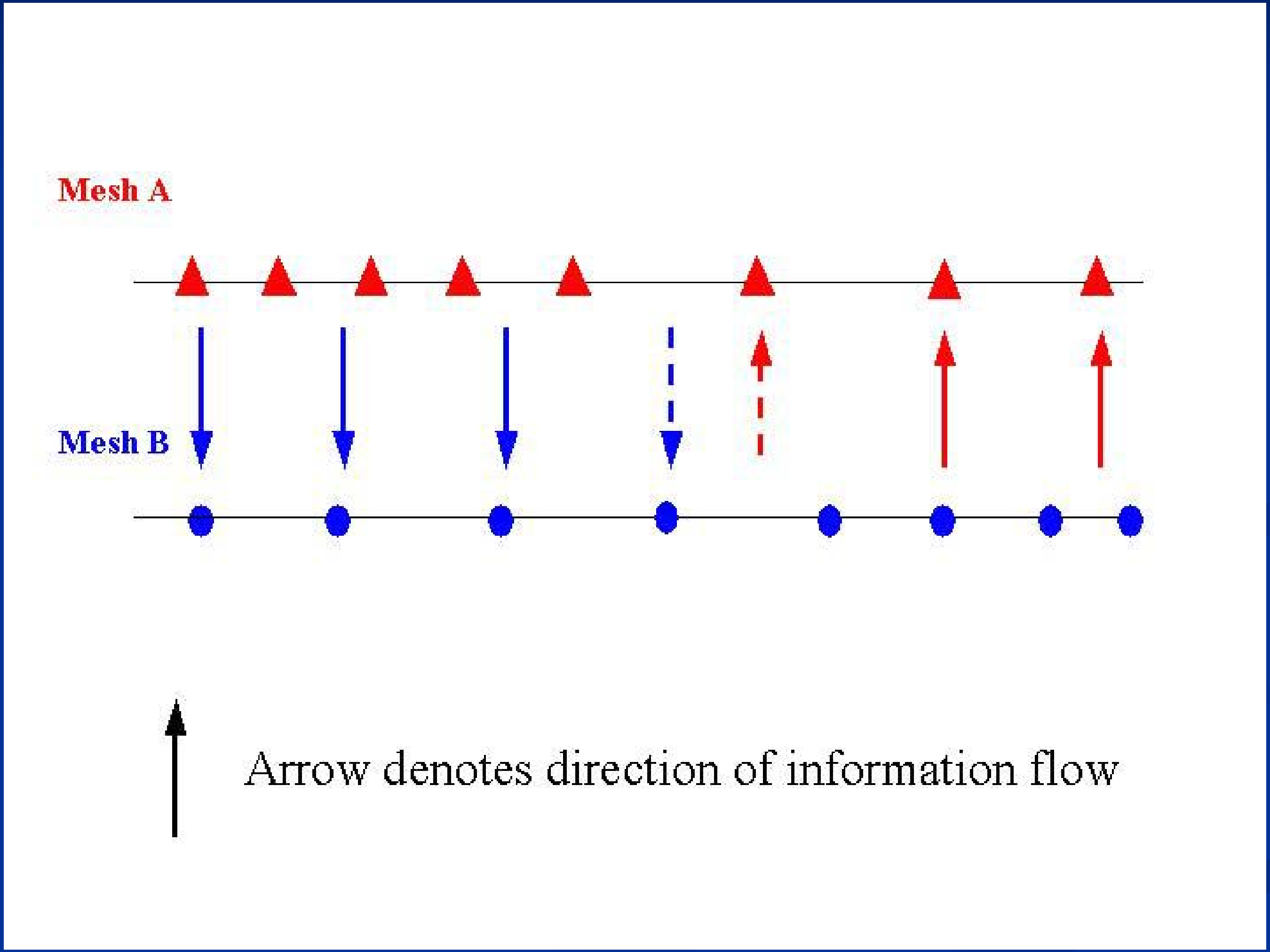
Level-2 Interpolation

- Optimization of overlap
- Interpolation points added after to Level-1 Interpolation
- Has effect of expanding the automatically-cut holes and shrinking the outer edges of overlapping zones
- Finest grid points remain active interior points
- Coarser grid points are interpolated from available donor cells of finer neighboring zones
- Methodology is robust, requires no user inputs, and maximizes communication between overlapping zones

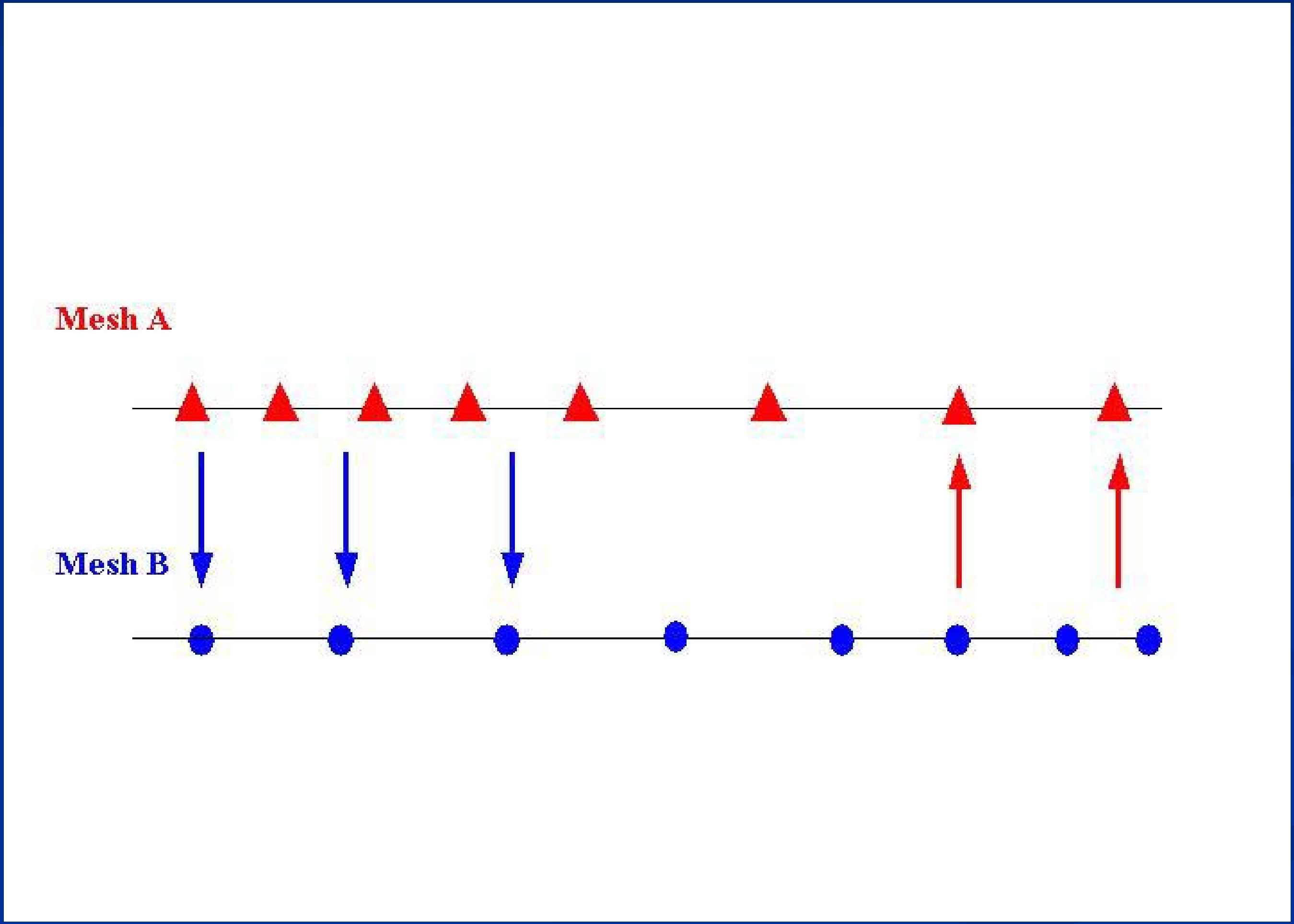
Level-2 Interpolation One Dimensional Example



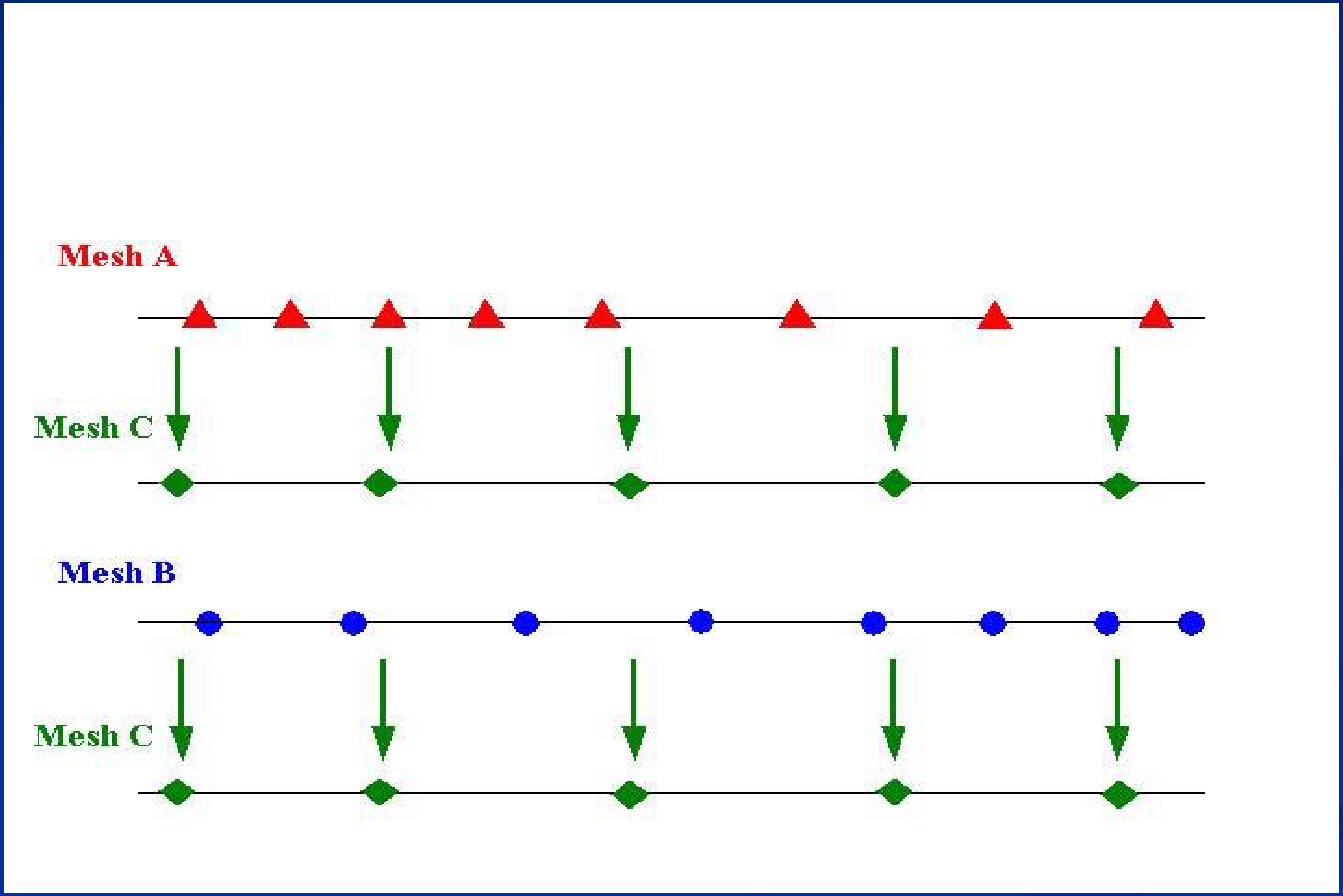
Step 1: Interpolate Between Meshes



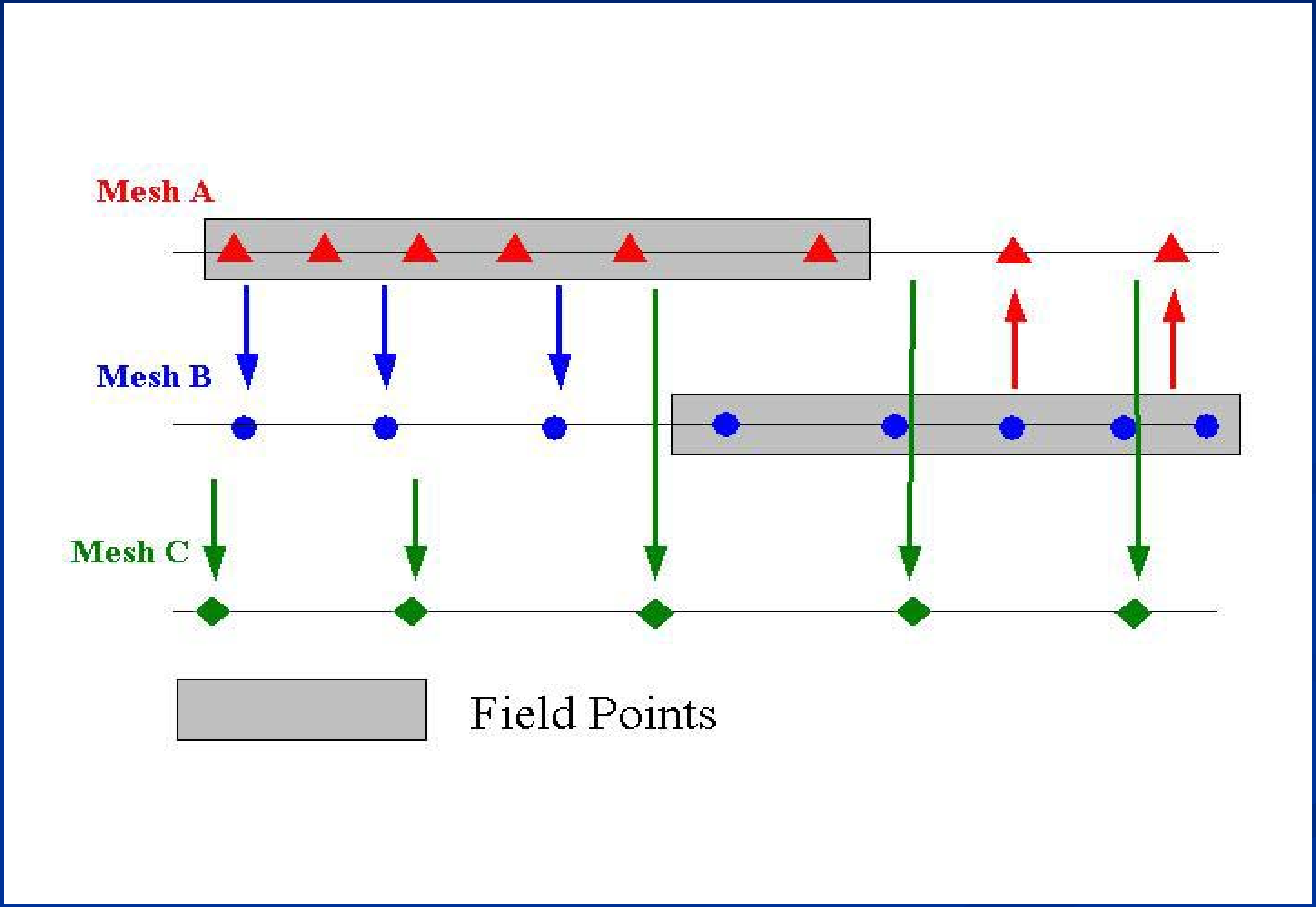
Step 2: Remove Invalid Interpolations



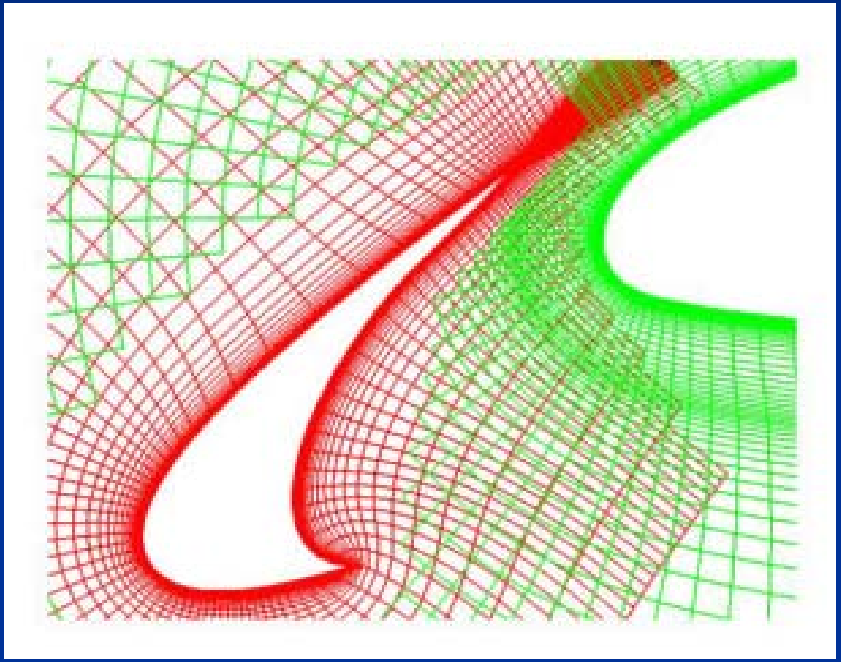
Repeat Step 1 and 2 for Other Meshes

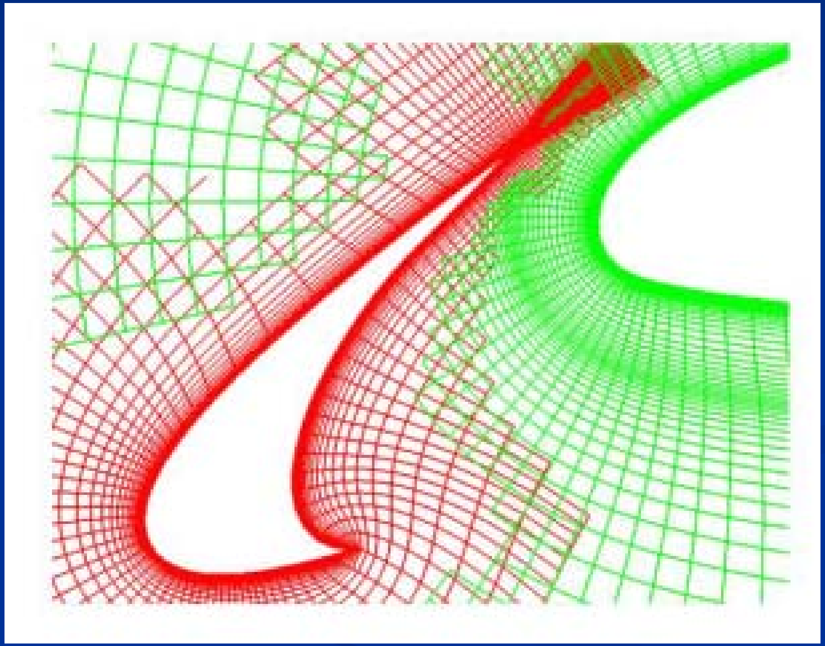


Step 3: Keep Finest Mesh Points



Optimized Overlap Example

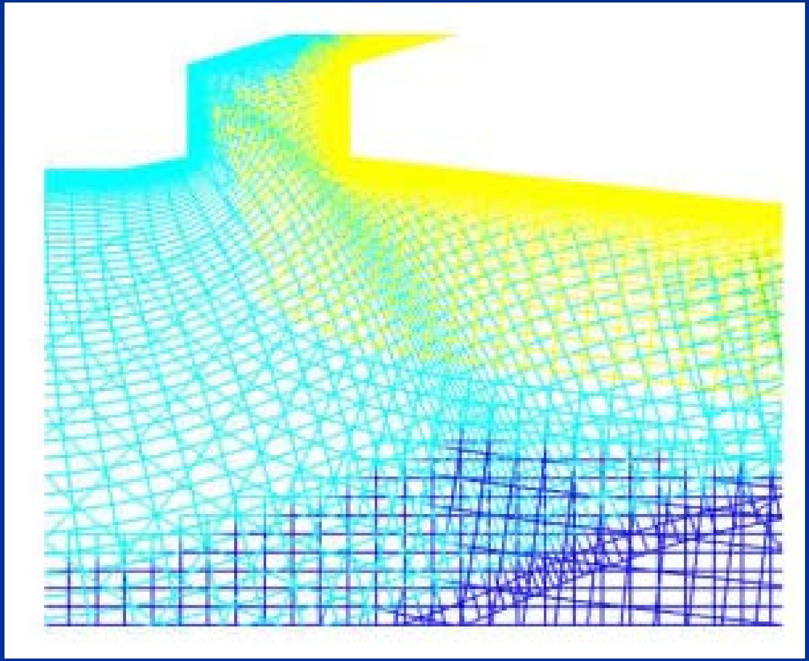


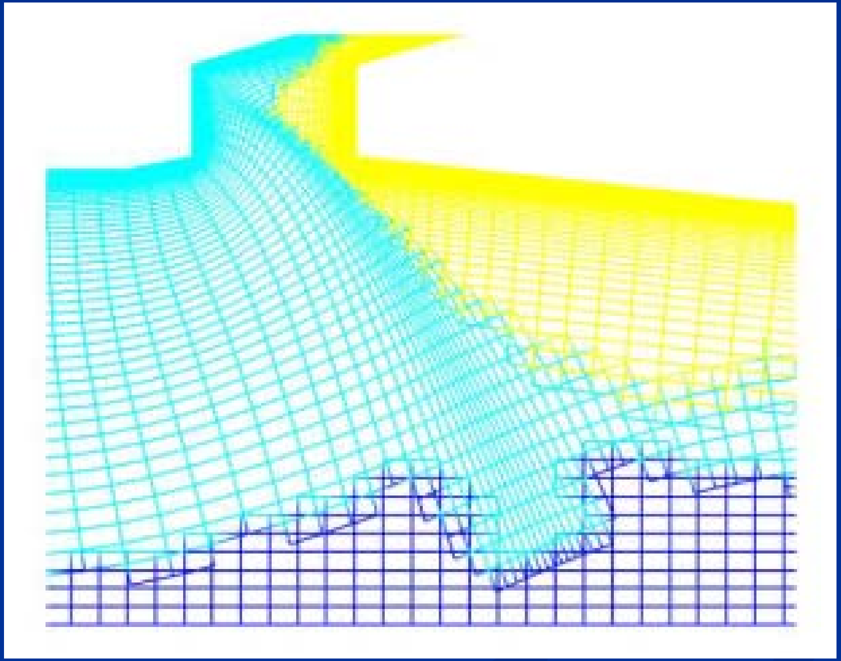


Non-Optimized Overlap

Optimized Overlap

Optimized Overlap Example

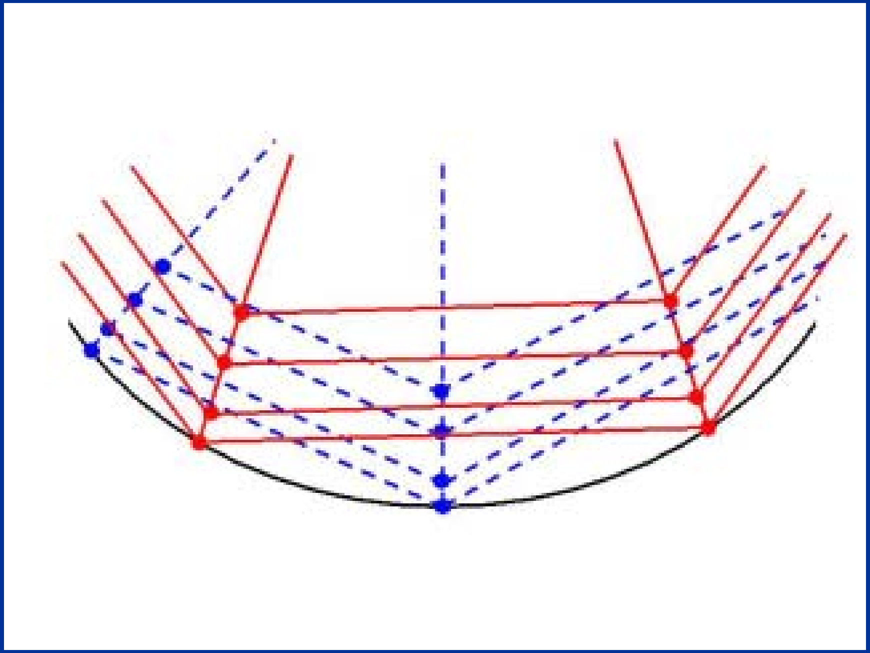

Non-Optimized Overlap

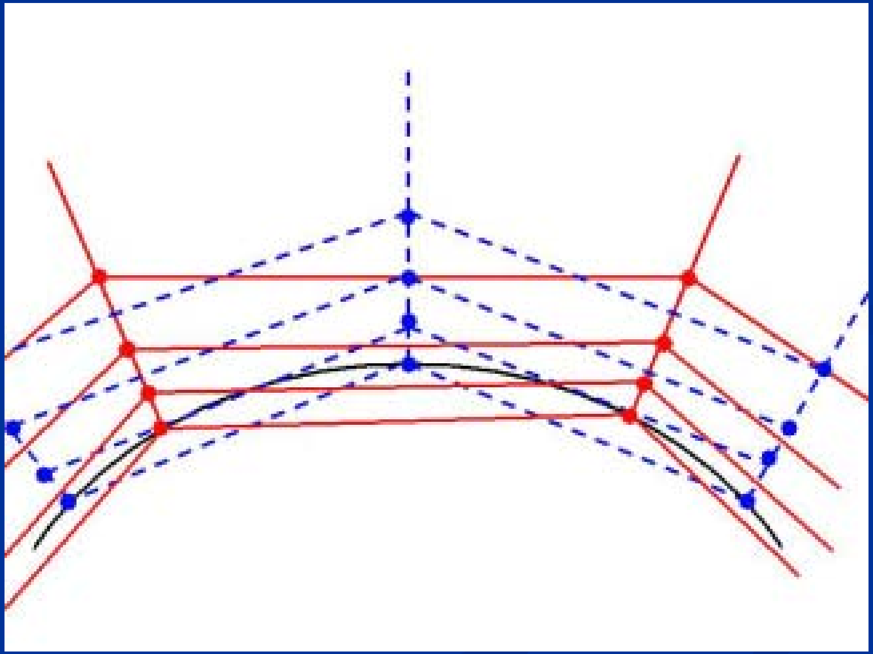

Optimized Overlap

Projection

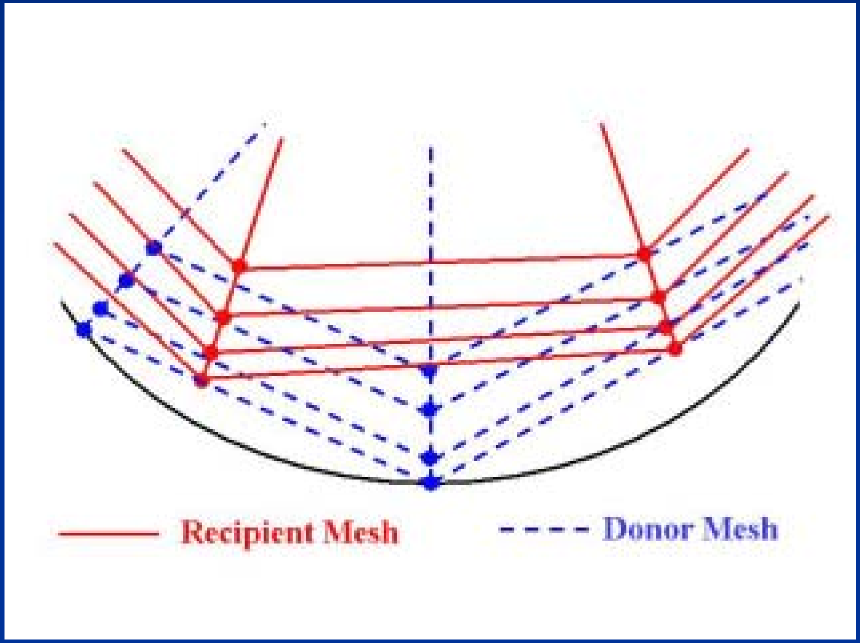
- Corrects interpolation problems that may occur on curved viscous surfaces
  - Cell aspect ratio > 1000 near viscous surface is typical
- Pegasus 5 projection step alters interpolation coefficients, not actual grid points
- Projection is performed internally and typically requires no user input

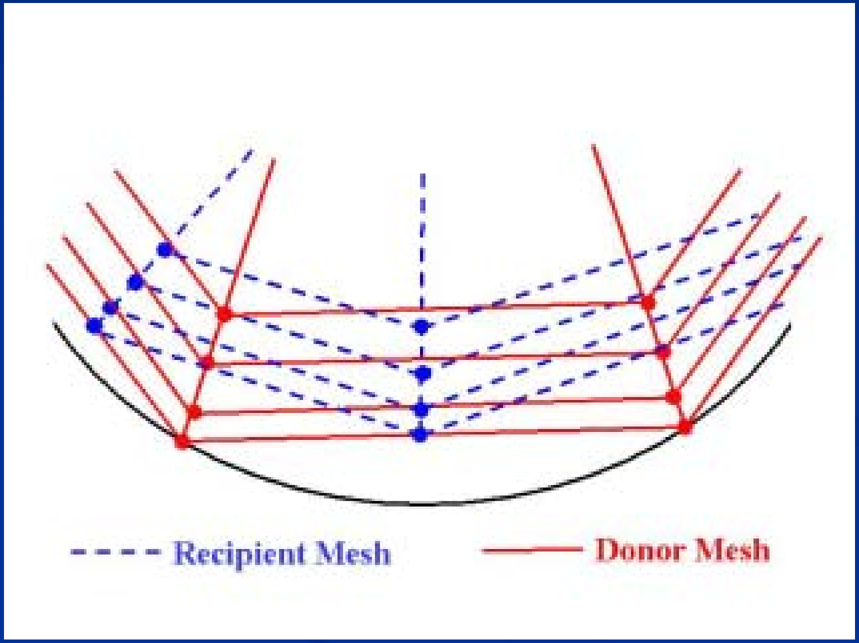
Problem: Linear Discretization on Curved Surfaces


Concave Surface


Convex Surface

Solution: Projection





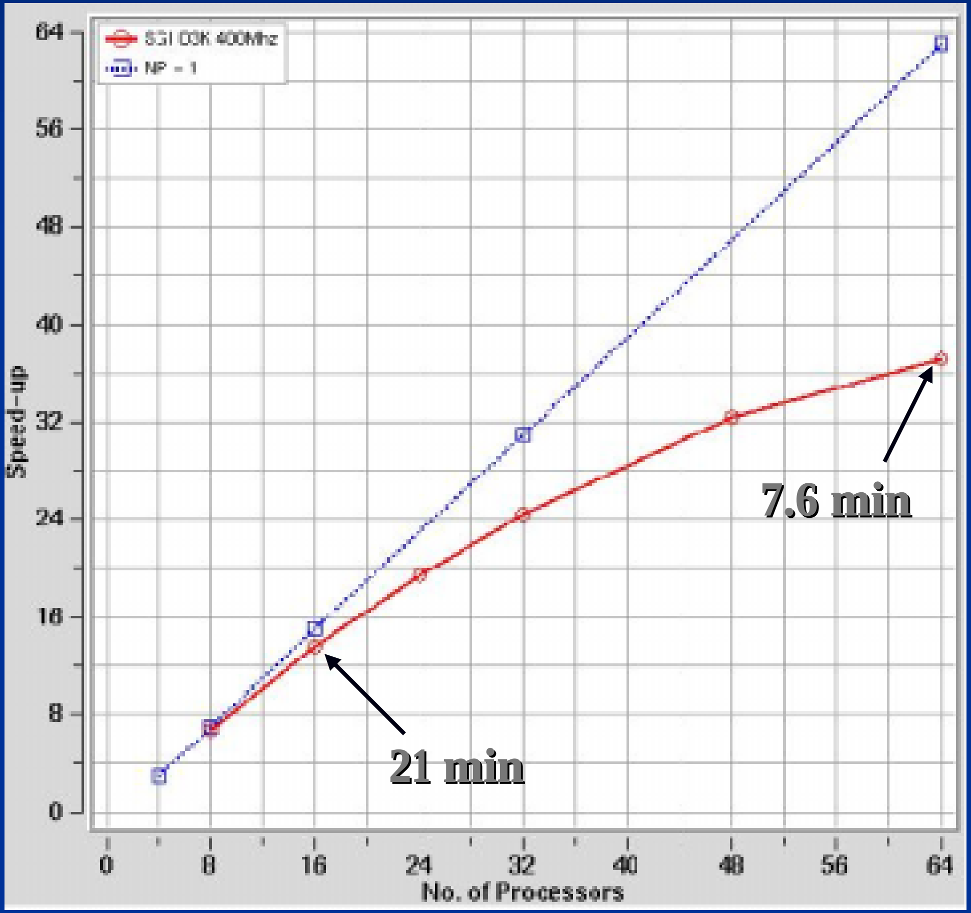
Points are Projected for Interpolation Only
Original Meshes are Retained

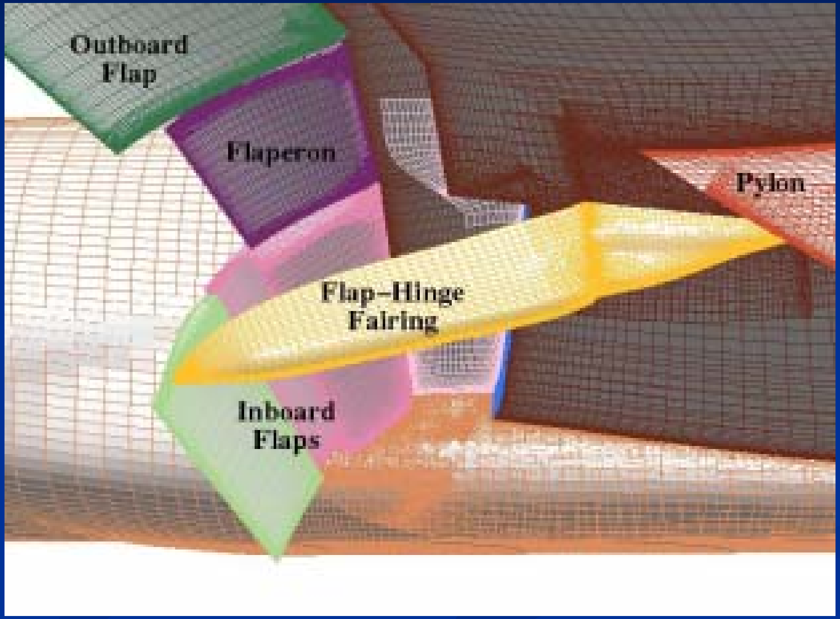
Parallelization

- Code is composed of many tasks
  - Projection, ADT, interpolation, hole-cutting, level-1 interpolation, level-2 interpolation, etc
  - Most tasks are independent of each other
  - Each task reads its input from disk files and write their results to disk files
- Parallelization uses Message-Passing-Interface (MPI)
  - One master process to monitor and distribute the work
  - Many worker processes, one per CPU
- Reliably reproduces results of serial code
- The larger the grid system, the better the parallel scaling

Parallel Speedup: Boeing 777-200 Example

SGI Origin: Total CPU time 283 min

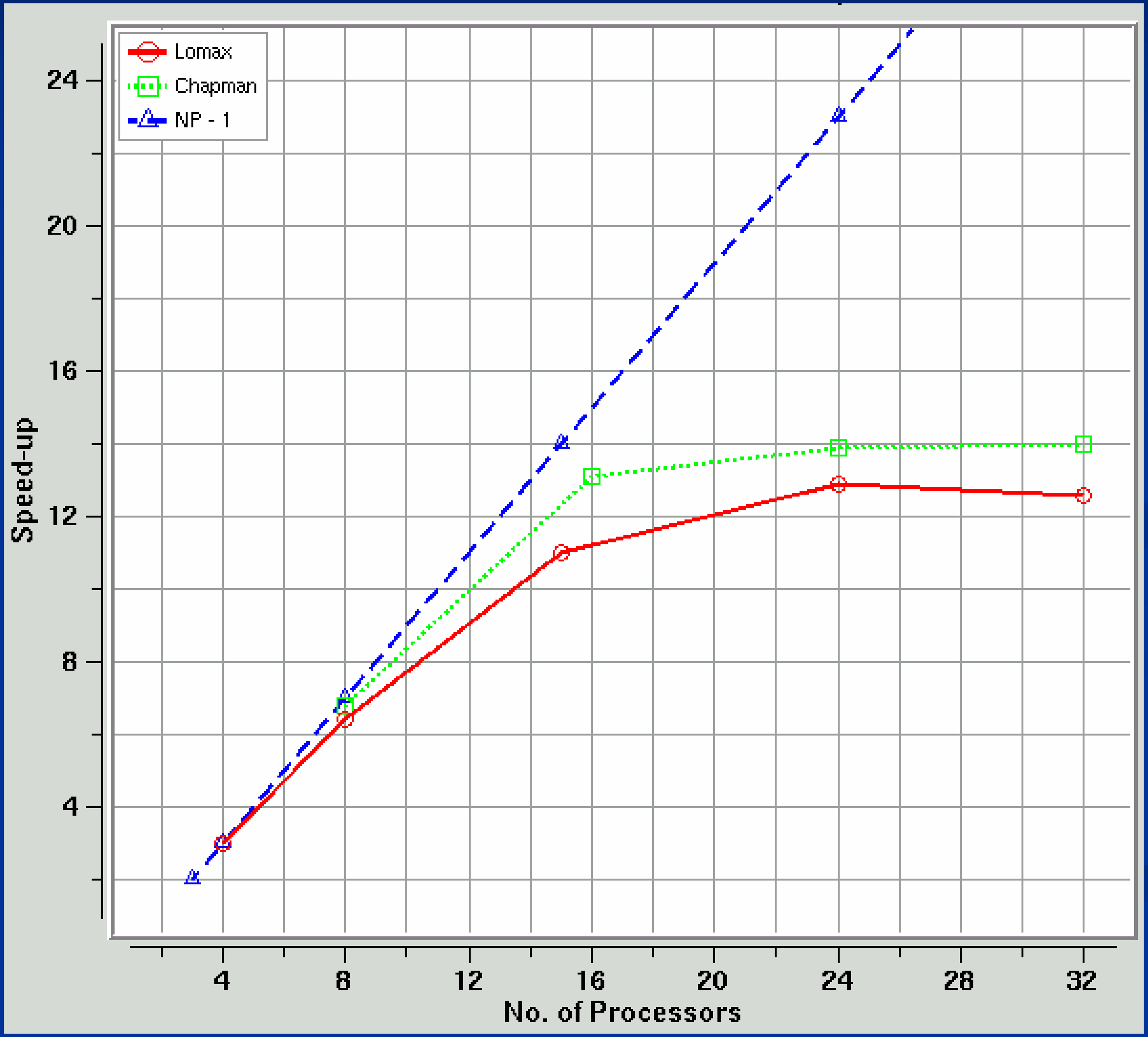




22 million points
79 zones

PEGASUS5 Parallel Scaling

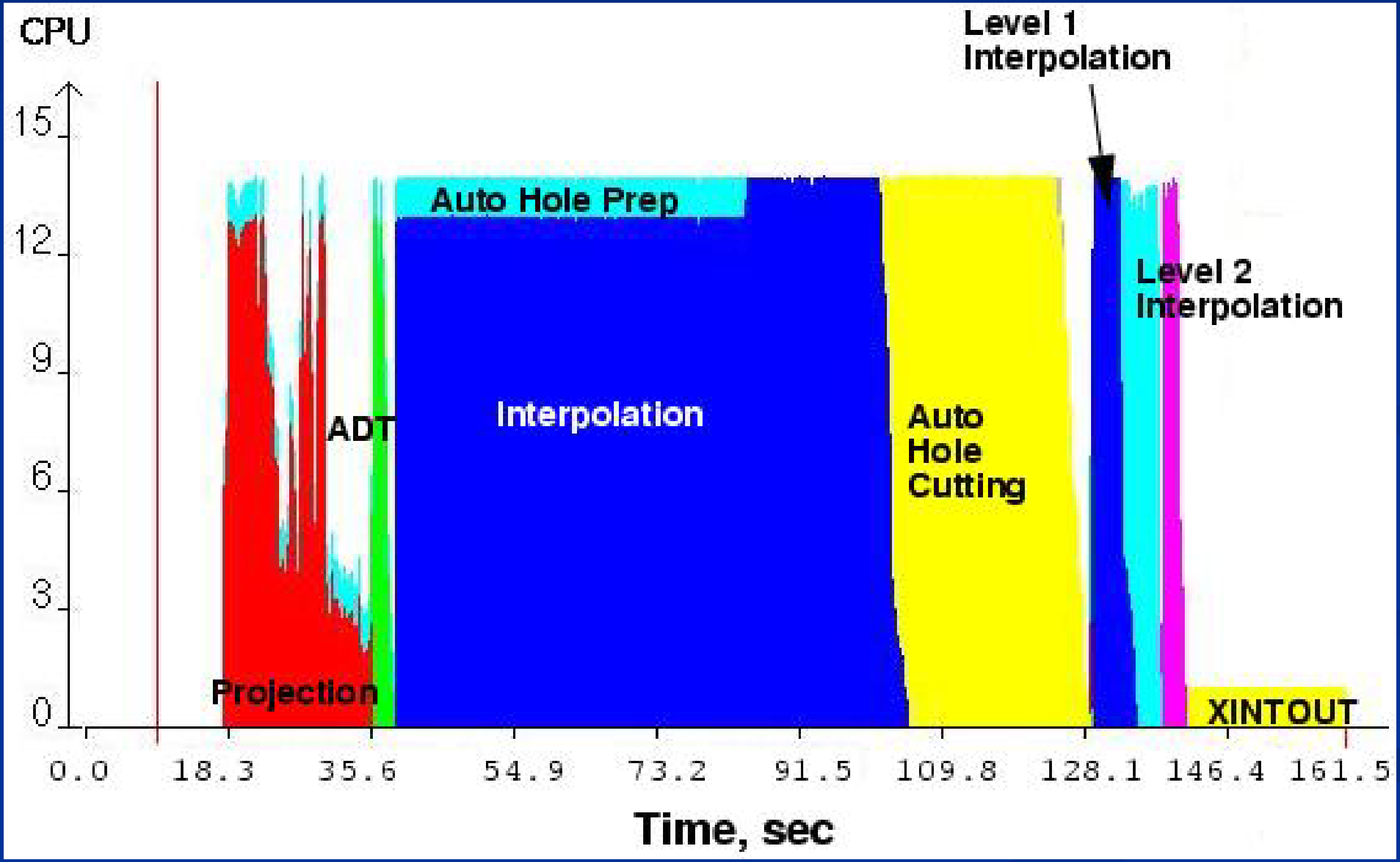
Harrier grid system: 52 zones, 2.5 million grid points



PEGASUS 5 Parallelization

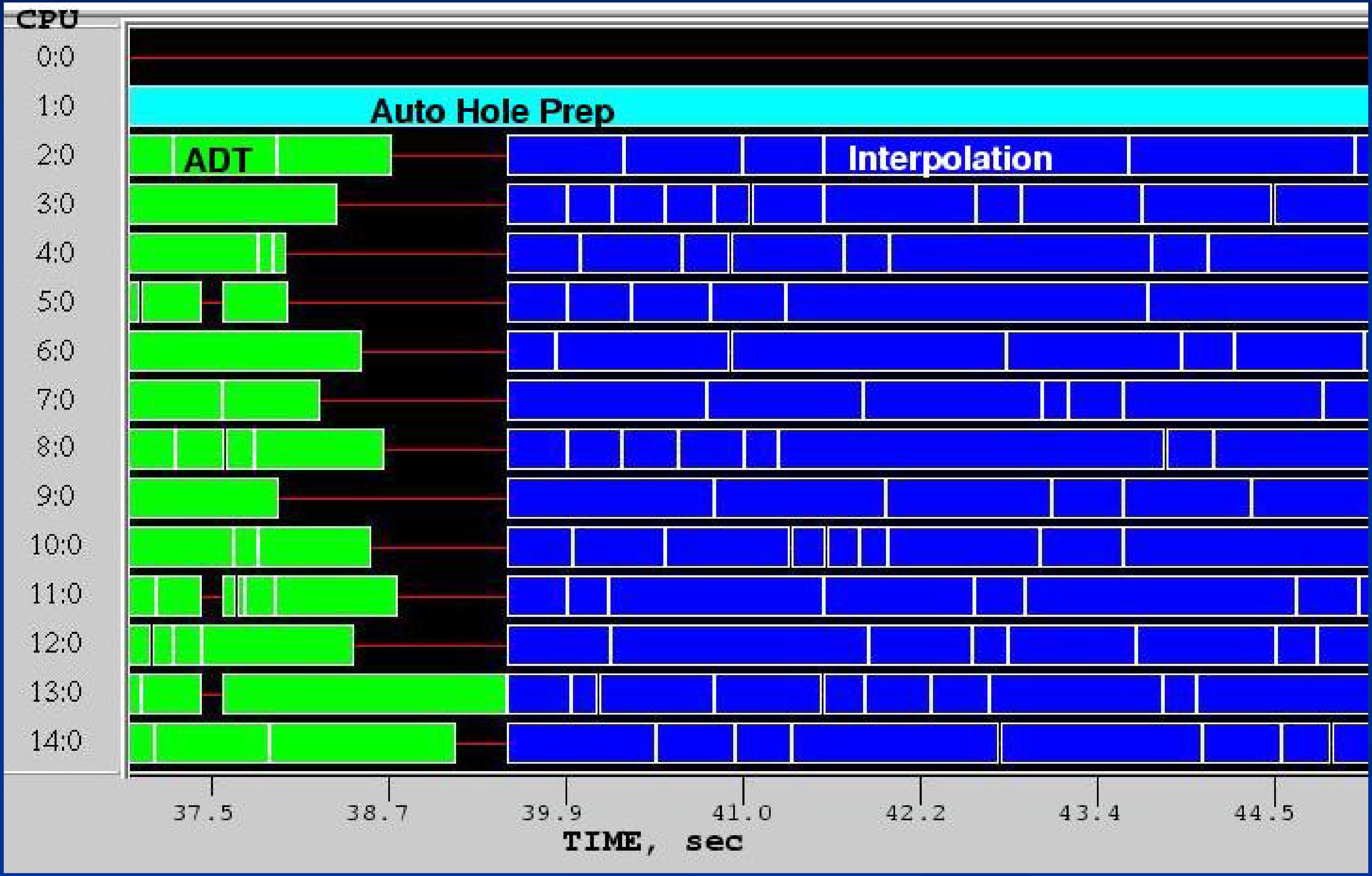
15 Processors on an SGI O2K

Harrier grid system: 52 zones, 2.5 million grid points



PEGASUS 5 Parallelization

Parallel execution: barrier between ADT and INTERPOLATION



Restarting

- Pegasus5 execution consists of many individual tasks
- Each task has a defined set of dependencies (inputs) that are stored to a disk file
- Each task results in one or more output files stored to a disk file
- Automatically determines which tasks are out of date based on internal time-stamps of input and output disk files
  - Internal time-stamps written as first and last record of each disk file
  - An incomplete or inconsistent file is considered out of date
- Upon execution pegasus5 firsts checks all files and determines which tasks need to be run
- Can successfully restart for:
  - Modifications to user inputs or zones
  - Addition of new zones
  - Incomplete previous run or computer crash
- Allows incremental buildup of complex configurations

Pegasus5 Inputs

- Input requirements:
  - Standard input file, namelist format
  - Volume grids in individual files:

    X_DIR/zonename1.x, …. , X_DIR/zonenameN.x
- Tools to assist in generating these inputs
  - peg_setup script
    - Requires Overflow input file and multi-zone plot3d grid file containing all of the volume grids
  - Chimera Grid Tools scripts: BuildPeg5i

Pegasus5 Input File Example

Set double fringe

OFFSET used to expand auto hole

```
$GLOBAL
   FRINGE = 2,
   OFFSET = 1,
   $END

$MESH NAME =       'body', KINCLUDE= 2, -2, LINCLUDE= 2, -1
                           OFFSET=2, $END

$MESH NAME = 'bodynose', JINCLUDE= 2, -1, LINCLUDE= 2, -1, $END

$MESH NAME =       'wing', $END

$MESH NAME =   'wingcap', $END

$MESH NAME =   'wingcol', $END
.
.
.
```

Pegasus5 Input File Example

```
$BCINP ISPARTOF =      'body',
    IBTYP =      5,    17,    17,    15,
    IBDIR =      3,     2,    -2,    -1,
    JBCS  =      1,     1,     1,    -1,
    JBCE  =     -1,    -1,    -1,    -1,
    KBCS  =      1,     1,    -1,     1,
    KBCE  =     -1,     1,    -1,    -1,
    LBCS  =      1,     1,     1,     1,
    LBCE  =      1,    -1,    -1,    -1,
    YSYM  = 1,
    $END
.
.
.
```

4 BCs (columns)
5 = viscous wall
17 = symmetry
15 = axis

Symmetry in Y

Pegasus5 Execution

- Once you have the input file and the volume grids are installed in the X_DIR directory you can execute the code:
  - Serial version:

    ```
    pegasus5 < peg.i
    ```
  - MPI Parallel version using $NCPUS cpus:

    ```
    mpiexec –NP $NCPUS pegasus5mpi < peg.i
    ```
- Note: mpi version requires that all CPUs have access to the same copy of the working directory

Pegasus5 Output

- Pegasus5 creates a directory named `WORK` which contains all of the intermediate output files created by each internal task
  - Typically no need to examine or read these files directly
  - In order to re-run a case from the beginning, simply remove `WORK`
- All informational output written to a file named `log.mmdd.hhmm`. Examine this file to see what Pegasus5 did
  - Note: during execution the mpi version will create multiple log files which will be concatenated together upon successful completion of the run

```
log.mmdd.hhmm.0000
log.mmdd.hhmm.0001
log.mmdd.hhmm.0002
…
log.mmdd.hhmm.0015
```

- The `XINTOUT` file contains all of the interpolation stencils and blanking information used by the flow solver

Post Execution

- Examine log file and verify successful completion
- Examine minimum hole cuts and make sure no active points are left inside a solid body
  - Plot hole boundaries in plot3d
  - Plot grid slices in overgrid, tecplot, fieldview, etc
  - Look for orphan points left inside a solid body
- Examine and eliminate cause of orphan points

End of Log File: Stencils and Orphans

```
-------------------------------------------------------------------------------
                    |               |               |                         |
Mesh Name           |Interpolated   |Interpolation  |Orphan Points            |
                    |Boundary Points|Stencil        |                         |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
fuselage            |Level 1:  10634|Level 1:  32934|1st Fringe:      0       |
                    |Level 2:  24578|Level 2:  10498|2nd Fringe:      0(Fixed)|
                    |Total:    35212|Total:    43432|Total:           0       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
wing                |Level 1:  38609|Level 1:  30486|1st Fringe:      2       |
                    |Level 2:  49279|Level 2:  12261|2nd Fringe:      0(Fixed)|
                    |Total:    87888|Total:    42747|Total:           2       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
wingcap             |Level 1:  20251|Level 1:  12491|1st Fringe:      0       |
                    |Level 2:    242|Level 2:  22748|2nd Fringe:      0(Fixed)|
                    |Total:    20493|Total:    35239|Total:           0       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
fuselagefillet      |Level 1:   8827|Level 1:   9584|1st Fringe:      2       |
                    |Level 2:  14321|Level 2:  13172|2nd Fringe:      0(Fixed)|
                    |Total:    23148|Total:    22756|Total:           2       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
Grand Total         |Level 1: 262641|Level 1: 262641|1st Fringe:     14       |
                    |Level 2: 267467|Level 2: 267467|2nd Fringe:      0       |
                    |Total:   530108|Total:   530108|Total:          14       |
                    |               |               |                         |
-------------------------------------------------------------------------------
```

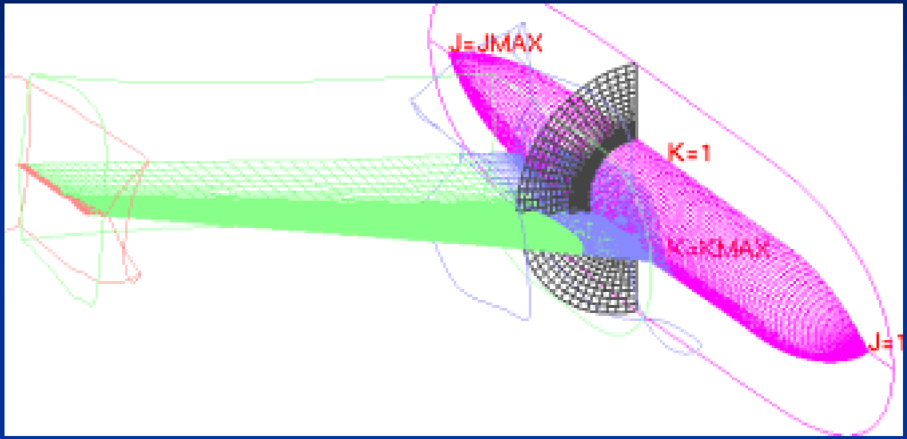
End of log file: Execution Time

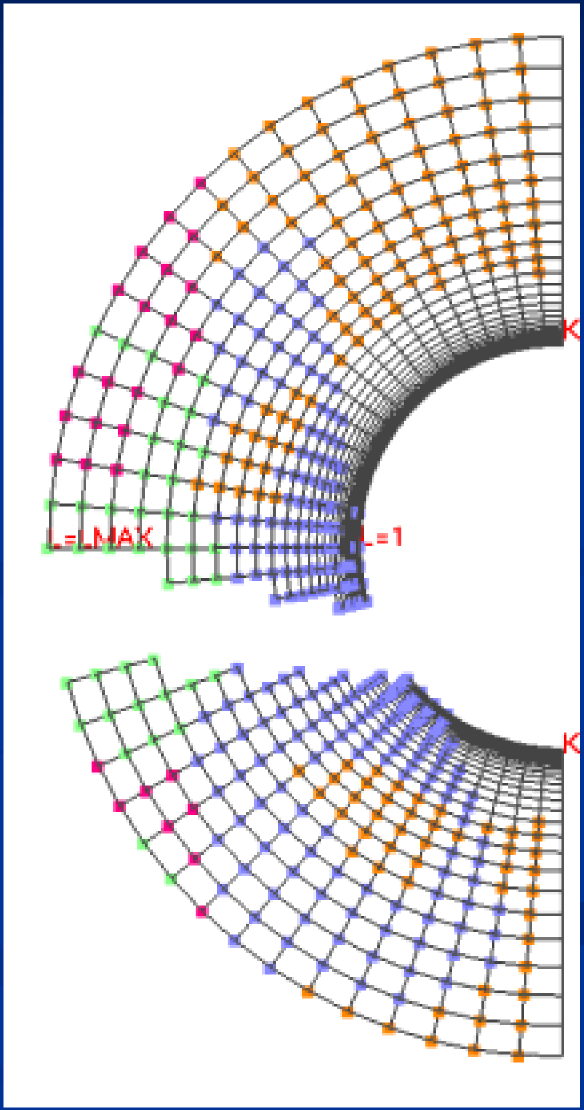
```
 PROCESS                 CPU(sec)       WALL(sec)   Sub-procs   Max sub-proc(sec)

projection                 13.875           2.328        122           1.672
adt                         4.656           1.266         13           0.906
interpolate                65.922          24.586        122           3.867
auto_hbound                66.438          26.898          3          26.906
man_hbound                  0.000           0.000          0           0.000
auto_cut                   42.234           4.906         30           4.805
man_cut                     0.000           0.000          0           0.000
comp_hole                   1.156           0.141         13           0.125
spec_int1                   0.508           0.055         13           0.062
spec_level1                 8.078           0.859         13           0.867
level1fix                   1.734           2.305          1           1.734
spec_int2                  19.859           2.195         61           0.930
spec_level2                 9.859           1.023         13           1.016
xintout                     1.469           1.477          1           1.469

 SUM of PROCESS TIME for all processes (secs):        235.789

 ELASPSED WALL TIME(secs):                         37.703

 EXECUTION SPEED-UP =       6.25 using    15 processors.
```

Output: `peg_plot`

- Grid file: use the `peg_plot` program to create the grid file used by the flow solver, and to plot and check the results of the Pegasus5 run
  - Use `peg_plot` option 3 first to examine the results of the hole cutting
  - The `peg_plot` options 1 and 2 blank out the higher-level fringes in the resulting grid file
    - This illustrates the borders of where information is passed between overlapping zones
    - Useful when plotting the flow solution as it minimizes the overlap
- Note: Overflow does not use the iblank array in the grid file, so any `peg_plot` option works when creating the grid file that will be passed to Overflow

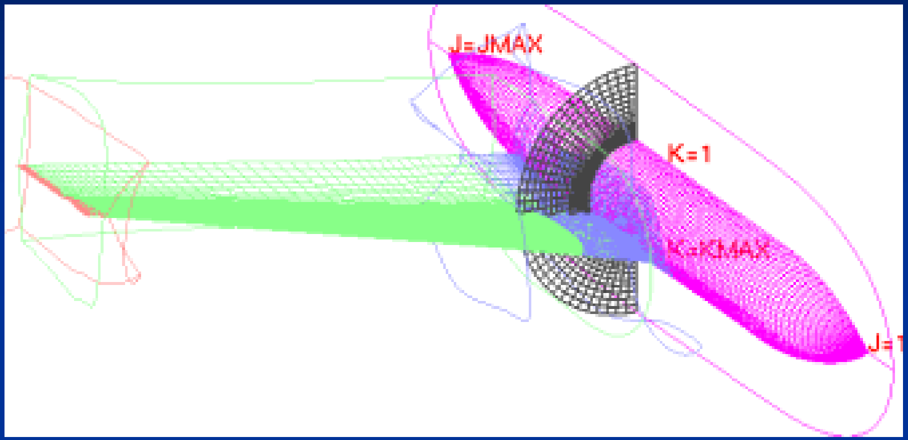
Example: `peg_plot 3`

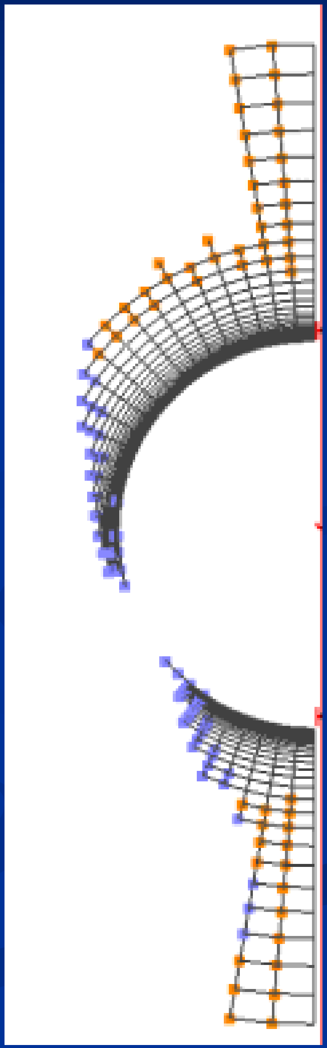




- Wing-body example using `peg_plot` option 3
- View the fuselage zone in `overgrid`
- Shows auto hole cut by the wing
- Fringe points shown with colored symbols

Example: peg_plot 2

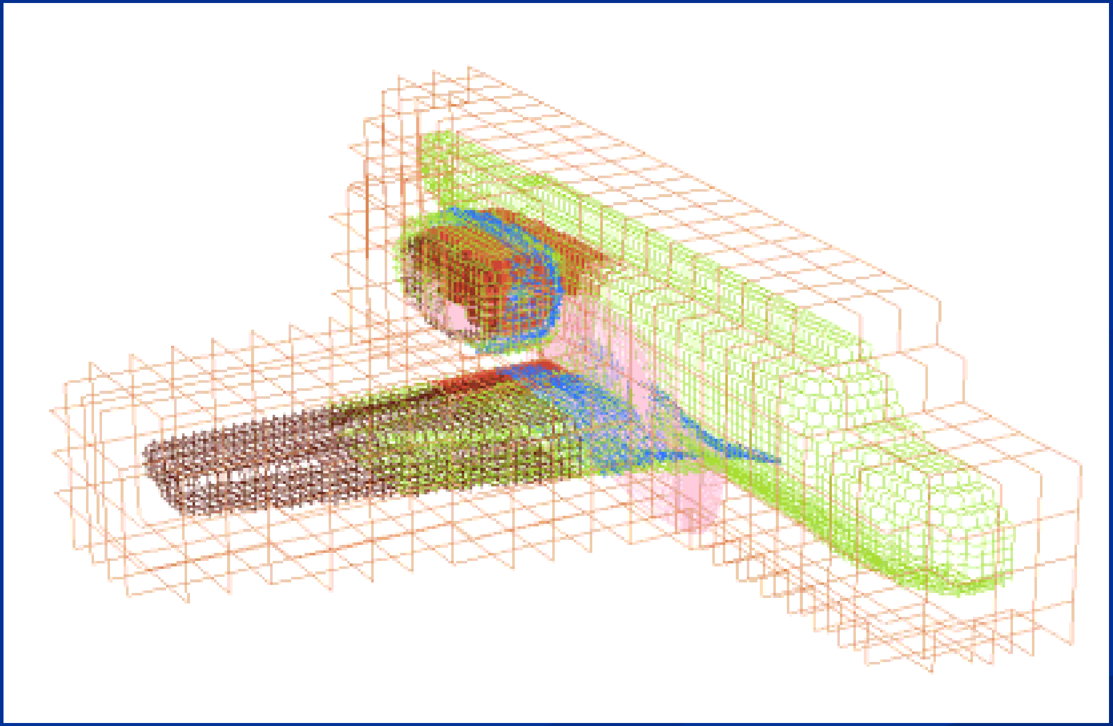




- Wing-body example using `peg_plot` option 2
- Higher-level fringe points have been blanked out
- Shows the virtual overlap after the Level-2 interpolation
- Flow-solver still keeps the higher-level fringes active: they can be used as donor cells for other zones

Examining the Hole Cuts

- Use `plot3d` function 2: plots the outlines of the holes
- Use `Overgrid, etc:` plot slices through grids
- Search log file for “composite hole”: lists number blanked points in each mesh
- Use `peg_hole_surf` to extract grid surfaces used by each `$HCUT` auto hole cutter



Custom Hole Cutting

- The `$HCUT` namelists are used to define separate auto hole-cutters
  - By default, no $HCUT namelist is included in input file, and pegasus5 uses ALL solid-wall surfaces to cut holes from ALL zones
  - Adding an $HCUT entry eliminates this

Solid walls of these zones must form a fully enclosed volume

List of zones which are cut

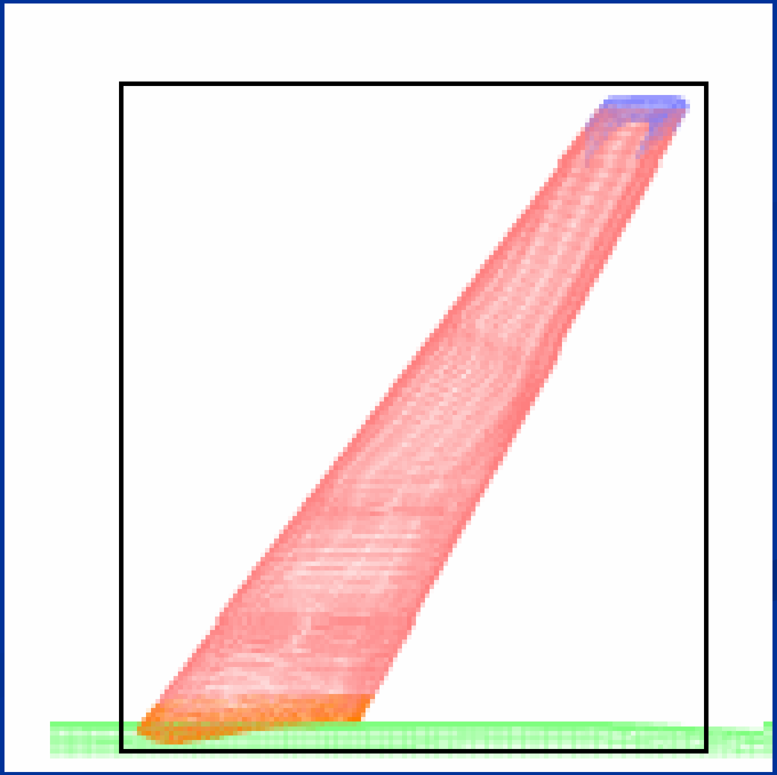
```
$HCUT NAME = 'hcutter1',
    MEMBER  = 'body1’, ‘body2',
    INCLUDE = 'bodynose', 'wing',
   wingcol',
    CNX = 512, CNY = 512, CNZ = 512,
    CARTX = -100.0, 100.0,
    CARTY = -50.0, 50.0,
    CARTZ = 0.0, 100.0,
    $END
```

Cartesian map dimensions

Bounding box of hole cutter

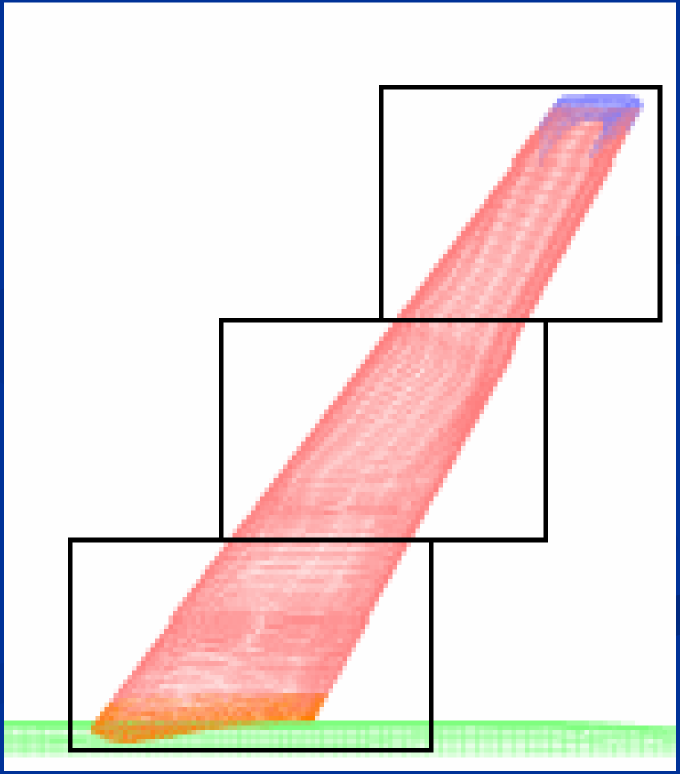
Custom Hole Cutting

```
$HCUT NAME = ‘winghole',
    MEMBER  = ‘wing’, ‘wingcol', ‘wingcap', ‘body',
    INCLUDE = 'body', 'wingbox', ‘bodybox', ‘farbox’,
    CARTX = 100.0, 400.0,
    CARTY = 10.0, 150.0,
    $END
```



Custom Hole Cutting

```
$HCUT NAME = ‘winghole1',
    MEMBER  = ‘wing’, ‘wingcol', ‘body',
    INCLUDE = 'body', 'wingbox', ‘bodybox', ‘farbox’,
    CARTX = 100.0, 250.0,
    CARTY = 10.0, 51.0,
    $END
$HCUT NAME = ‘winghole2',
    MEMBER  = ‘wing’,
    INCLUDE = 'wingbox', ‘farbox’,
    CARTX = 200.0, 350.0,
    CARTY = 50.0, 101.0,
    $END
$HCUT NAME = ‘winghole3',
    MEMBER  = ‘wing’, ‘wingcap',
    INCLUDE = 'wingbox', ‘farbox’,
    CARTX = 240.0, 400.0,
    CARTY = 100.0, 150.0,
    $END
```



Hole-Cutting Troubleshooting

- No holes cut due to leak or gap in solid-wall surfaces
  - Use `CARTX,CARTY,CARTZ` to seal gap
  - Use PHANTOM zone to seal gap
  - Edit input file and extend solid-wall boundary
- Holes too small near thin bodies, such as TE of a thin wing:
  - Increase `OFFSET` to enlarge holes
  - Increase `CNX, CNY, CNZ` to improve resolution of Cartesian map
- Hole points not cut properly near collar grids
  - Increase `OFFSET` to enlarge holes
- Holes cut at zone edges adjacent to solid walls in regions of high curvature – can occur with inadequate resolution relative to curvature
  - Increase grid resolution
  - Use `$REGION` and `$VOLUME` namelists to "unblank" holes

Manual Hole Cutting

- Manual hole-cutting functionality from pegsus4 has been retained in Pegasus5
  - Can be used as an additional tool to refine holes
- `$BOUNDARY/$SURFACE` namelists
  - Cans specify a group of zonal surfaces which will cut holes in the specfied zones
- `$BOUNDARY/$BOX` namelists
  - Can specify range of x,y,z coordinates of a box which will cut holes in the specified zones
- `$REGION/$VOLUME` namelists
  - Can specify range of j,k,l zonal indices to create a hole or to unblank part of an existing hole

Orphan Points

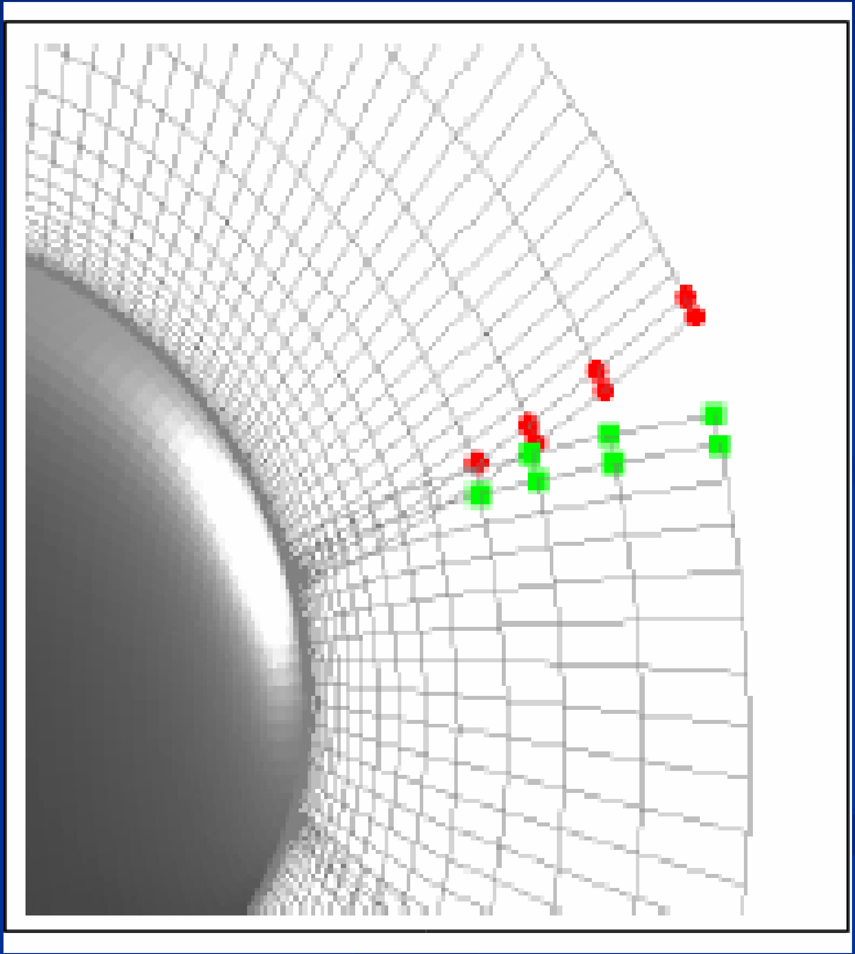
- Orphan points are fringe points for which no valid interpolation donor can be found
- Orphans are reported in the log file, in the output of `peg_plot`, and using the `peg_orph` program
- 2nd-level fringe-point orphans are reset to active interior points
- Overflow will update the solution data at orphan points by averaging the neighboring grid points
  - A few isolated orphan points are usually acceptable, but it is advisable to find and fix most or all orphans
  - Orphans on solid-wall surfaces usually indicate a serious problem with surface resolution or projection, and should be fixed
- Plot orphans using the `plot3d` or `overgrid` programs

Causes of Orphans

- Bad hole cuts
- Insufficient overlap
- Poorly resolved geometry in regions of surface curvature
- Inappropriate or missing boundary conditions

Insufficient Overlap

- Possible fixes:
  - Increase surface-grid overlap
  - Add more splaying to the boundary conditions in `hypgen`
  - Add a cartesian box grid to resolve the open space



Utility Codes

- `peg_setup`: creates
- `peg_hole_surf`: creates plot3d grid files containing surfaces of each $HCUT entry
- `peg_plot`: creates composite plot3d grid file
- `peg_diag`: produces diagnostic file for plotting interpolation parameters and connection info
- `peg_orph`: lists orphan points for each zone
- `peg_proj`: creates diagnostic plot3d files for projection
- `XINtegrity`: verifies integrity of the XINTOUT file

Summary

- Pegasus5 successfully automates most of the oversetting process
  - Dramatic reduction in user input over previous generations of overset software
  - Reduced complex-geometry oversetting time from weeks to hours
  - Significant reduction in user-expertise requirements
- Not a "black-box" procedure: care must be taken to examine the resulting grid system
- Additional documentation and examples available online:

  `http://people.nas.nasa.gov/~rogers/pegasus/status.html`

Nomenclature

- **Grid System:** A collection of zones together with boundary conditions and connectivity data ready to input into a flow solver
- **Zone:** a single structured grid composed of ordered grid points
- **Cell:** a hexahedron composed of 8 grid points and 6 faces
- **Grid point:** a single point in a zone identified uniquely by its j,k,l indices
- **Fringe point:** a grid point which will be updated in the flow-solver via interpolation of data from a neighboring zone
- **Outer-boundary fringe point:** a fringe point on the boundary edge of a zone
- **Hole-boundary fringe point:** a fringe point adjacent to a hole point
- **Hole point:** a grid point which has said to have been “blanked out” and whose data will not be used by the flow solver
- **Orphan point:** a fringe point for which a valid donor cell cannot be found
- **Interior point:** a grid point which does not lie on the zonal boundary
- **Iblank value:** each grid point is assigned an integer value to denote type of point:
  - iblank<0: fringe point
  - iblank=0: hole point
  - iblank=1: active interior point
  - iblank=101: orphan point